Mar 80

MULTI-ICE™ SOFTWARE MULTIPLE-IN-CIRCUIT-EMULATOR

Facilitates software and hardware debugging of multi-processor systems.

Allows two In-Circuit Emulators to operate simultaneously in a single Intellec Microcomputer Development System.

Provides enhanced software features: Symbolic Display of Addresses, Macro Commands, Compound Commands, Software Synchronization of Processes, and INCLUDE File Capability.

Supports In-Circuit Emulator combinations, 85/85 Emulators and 85/49 Emulators (ICE-49 Emulator supports the design using MCS-48 chip family).

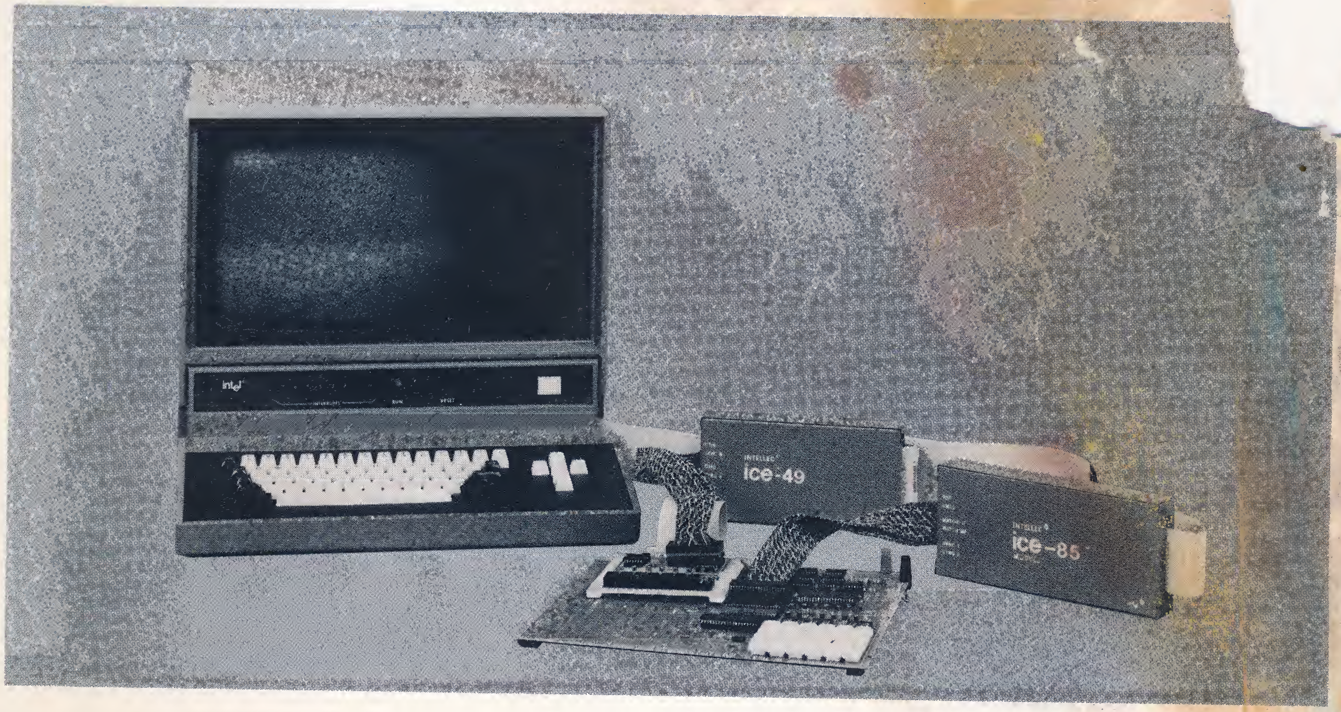
Functions under the supervision of ISIS-II Disk Operating System.

Supports ICE-85 Emulator Hold Request/Hold Acknowledgement ha shake while in both emulation and i rogation modes. (Can be used for Dynamic RAM refresh.)

Multi-ICE In-Circuit Emulator is a software product which allows two Intel In-Circuit Emulators to run simultane in a single Intellec Microcomputer Development System. Multi-ICE software used in lieu of the standard ICE sc gives users full control of the Intellec Microcomputer Development System, and the two ICE modules for hard and software debugging of multi-processor systems.

Enhancement features available with Multi-ICE software include a compound command ca [illegible] user to "program" a diagnostic or exercise sequence. Also included are repeat and co commands, and the ability to invoke the macro commands by name.

A special EPROM set for the ICE-85 Emulator is included. The new firmware will enable the ICE-8 Hold-Request and Hold-Acknowledgement hand-shake protocol both while in emulation an mode. This allows the ICE-85 Emulator to support typical dynamic RAM and DMA applications.

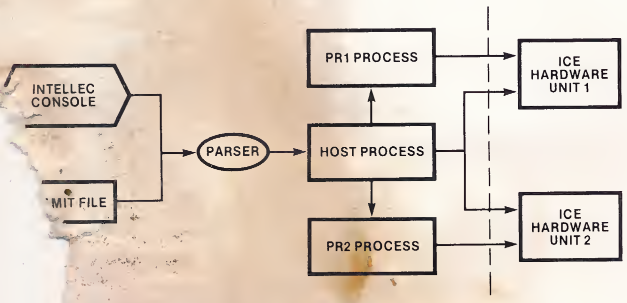


[illegible]

MULTI-ICE OPERATION

Multi-ICE software is a debug tool which allows two ICE emulators to begin and stop in sequence. Once started, two ICE emulators emulate simultaneously and independently. Thus, Multi-ICE software permits the debugging of asynchronous or synchronous multi-processor systems.

A conceptual model for the Multi-ICE software can be illustrated with the following block diagram.



Block Dia-ram of Multi-ICE^{TM} Operation

e thre [illegible] sses [illegible] the Multi-ICE environ- [illegible] the Host [illegible] and the two ICE processes to [illegible] l the two [illegible] vare modules. The processor [illegible] se three [illegible] the microcomputer in the [illegible] Micro [illegible] pment System. Only the [illegible] n Multi-ICE software is [illegible] aces with the console, [illegible] om the console or from a file, [illegible] termediate code, and loads the [illegible] mand code buffer or ICE com- [illegible]

[illegible] utes commands from its com- [illegible] using the execution software and [illegible] he Ho [illegible] s current environment, either [illegible] r environment 2 (EN1 or EN2), as [illegible] EN [illegible] d EN2 are the operating environments [illegible] In-Circuit Emulators.

The user can change the execution environment (from EN1 to EN2 or vice versa) with the SWITCH command. Once the environment is selected, ICE operation is the same as with standard ICE software. In addition, the enhanced software capabilities are available to the user.

The two ICE processes (PR1 and PR2) execute commands from their command code buffers in their own environments (PR1 in EN1 and PR2 in EN2). The main functions of the two ICE execution processes are to control the operations of the two ICE hardware sets. The ACTIVATE command controls the execution of the ICE processes. Commands are passed on to each ICE unit to initiate the desired ICE functions.

The two ICE hardware units accept commands from the Host process or ICE processes. Once emulations start, the two ICE hardware sets will operate until a break condition is met or processing is interrupted by commands from the ICE execution processes.

ENHANCED DIAGNOSTIC SOFTWARE FUNCTIONS

Single ICE Operation

Multi-ICE software can be used for single ICE operation. The operating procedures will be identical to the Multi-ICE operation. All the enhanced software functions will be available. The performance will be the same as if the standard ICE software is being used.

Symbolic Display of Addresses

The user has the option of displaying a 16-bit address in the form of a symbol name or line number plus a hex number offset.

Macro Command

A macro is a set of commands which is given a name. Thus, a group of commands which is executed frequently may be defined as a macro. Each time the user wants to execute that group of commands, he may just invoke the macro by typing a colon followed by the macro name. Up to ten parameters may be passed to the macro.

Macro commands may be defined at the beginning of a debug session and then can be used throughout the whole session. If the user wants to save the macros for later use, he may use the PUT command to save the macro on diskette, or the user may edit the macro file off-line using the Intellec text editor. Later, the user may use the INCLUDE command to bring in the macro definition file that he created.

Example:

```
*DEFINE MACRO INITMEM      ;This macro clears the
                            memory and then loads the
                            programs.
.*SWITCH = EN1             ;Select environment 1 (ICE
                            Module 1)
.*BYTE 0 TO 100=0          ;Initialize memory to 0.
.*LOAD :F1:DRIVER          ;Load user program into
                            memory for ICE Module 1.
.*SWITCH = EN2             ;Select environment 2 (ICE
                            Module 2)
.*LOAD :F1:DR2             ;Load user program into
                            memory for ICE Module 2.
.*EM                       ;End of Macro
*                          ;To execute this Macro, user
                            types :INITMEM
```

Compound Command

Compound commands provide conditional execution of commands (IF Command) and execution of commands repeatedly until certain conditions are met (COUNT, REPEAT Commands).

Compound commands and Macro commands may be nested any number of times.

Example:

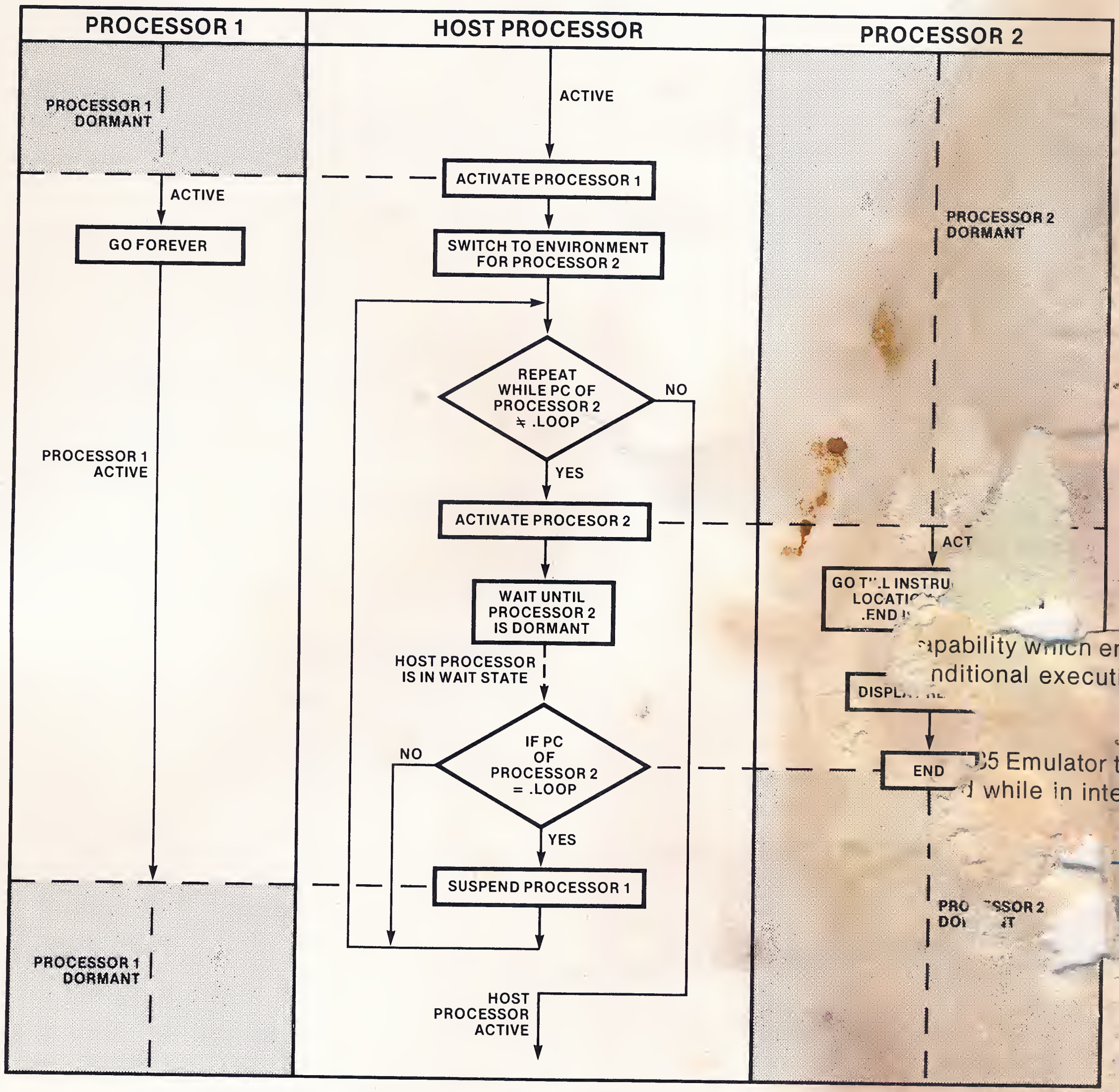
```
*DEFINE .I = 0             ;Define symbol .I to 0
*COUNT 100H                ;Repeat the following
                            commands 100H times.
.*IF .I AND 1 THEN         ;Check if .I is odd
..*BYT .I = .I             ;Fill the memory at location .I
                            to value .I
..*END
.*.I = .I+1                ;Increment .I by 1.
.*END                      ;Command executes upon
                            carriage-return after END
```

Software Synchronization of Processes

Up to three processes (Host, PR1 and PR2) can be active simultaneously in the system. An ICE process can be activated (ACTIVATE), suspended (SUSPEND), killed (KILL), or continued (CONTINUE). The Host process can wait for other processes to become dormant before it becomes active again. Through these synchronization commands, the user can create a system test file off-line yet be able to synchronize the three processes when the actual system test is executed.

Example:

The capability of the software synchronization commands is demonstrated by the following example. The flowchart shows the synchronization requirements. The program steps show the actual implementation.



Flowchart of the Example for Demonstrating Multi-ICE™ Synchronization Capability

```
*ACTIVATE PR1              ;Activate PR1
.*GO FROM 800              ;Start ICE Module 1
.*END                      ;End of Activate block
PR1 EMULATION BEGUN
*SWI=EN2                   ;Switch execution Environment to EN2
*REPEAT                    ;Repeat the following block of commands while PC is not equal to .Loop
.*WHILE PC < >.LOOP
.*ACT PR2                  ;Activate PR2
..*GO TILL .LOOP OR .END   ;Go till instruction at location .Loop or at location .END is executed
..*REGISTER                ;Display the registers
..*END                     ;End of Activate block
.*WAIT PR2                 ;Wait until PR2 is dormant
.*IF PC=.LOOP THEN
..*SUSPEND PR1
..*END                     ;End of IF block
.*END                      ;End of REPEAT block
*
```

INCLUDE File Capability

The INCLUDE command causes input to be taken from the file specified until the end of the file is encountered, at which point, input continues to be taken from the previous source. Nesting of INCLUDES is permitted. Since the command code file can be complex, the ability to edit offline becomes desirable. The INCLUDE command allows the user to pull in command code files and Macro commands created offline which can then be used for the particular debugging session.

Example:

```
*INCLUDE :F1:PROG1          ;Cause input to be taken
                             from file PROG1
*MAP 0 LENGTH 64K=USER      ;Contents of the file PROG1
                             are listed on screen as they
                             are executed.
*MAP IO 0 TO FF = USER
*SWITCH = EN2
*LOAD :F2:LED.HEX
*SWITCH = EN1               ;End of the file PROG1
*                           ;After the end of file is
                             reached, control is returned
                             to console.
```

SPECIFICATIONS

Equipment Supplied:

- Multi-ICE Flexible diskettes
  (one each in single and double density)
  Contains software that supports 85/85 Emulators and 85/49 Emulators

Special EPROM set for one ICE-85 Emulator

Operator's Manual

ILTI-ICE OPERATING ENVIRONMENT

equired Ha dware:

ellec Micr iter Development System

Model-800 88

Series II M , Model 230, and Expansion
  Chassis

Required Hardware: (Cont'd.)

64K bytes of RAM memory

Flexible disk drive(s)

—Single or double density

System Console

—CRT or hard copy interactive device

ICE-85 Emulator(s) or ICE-49 Emulator

Optional Hardware:

Printer

Additional flexible disk drives

Required Software:

Intel Systems Implementation Supervisor (ISIS-II)

ORDER NFORMATION:

Product	Description
MDS-350	lti-ICE Software

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, CA 95051 (408) 987-8080

Printed in U.S.A./H-98/0679/20K/BL/TP